Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Multi-objective artificial bee colony algorithm for energy-efficient scheduling of unrelated parallel batch processing machines with flexible preventive maintenance

Yarong Chen[a], Longlong Xu[b], Mudassar Rauf[c*], Pei Li[d*] and Jabir Mumtaz[c]

[a]*Professor, College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China*
[b]*Postgraduate Student, College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China*
[c]*Associate Professor, College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China*
[d]*Assistant Professor, College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China*

C H R O N I C L E

Article history:
Received January 15 2025
Received in Revised Format March 16 2025
Accepted April 17 2025
Available online April 17 2025

Keywords:
Artificial bee colony algorithm
Multi-objective optimization
Parallel batch-processing machine
Energy-efficient scheduling
Flexible preventive maintenance

A B S T R A C T

The parallel batch-processing machine scheduling problem is widely present in industries such as manufacturing, service, and healthcare, and becomes more complex when incorporating flexible preventive maintenance (FPM). This paper presents a mixed-integer programming (MIP) model and a multi-objective artificial bee colony (MOABC) algorithm to tackle the unrelated parallel batch-processing machine scheduling problem with flexible preventive maintenance (UPBPM-FPM). The objective is to simultaneously minimize the makespan, earliness and tardiness, and total energy consumption, providing a comprehensive solution to optimize both scheduling efficiency and energy use while incorporating preventive maintenance considerations. The MOABC algorithm integrates three key innovations: (1) a novel processing power-feature information (PP-FI) heuristic to generate high-quality initial solutions, (2) a hybrid selection strategy combining the hypervolume index and roulette wheel approach to improve diversity and convergence, and (3) a set of random and goal-oriented neighborhood search methods to enhance Pareto frontier. Experimental results demonstrate that the MOABC algorithm outperforms three classical algorithms, NSGA-III, ABC, and PSO, in terms of convergence, diversity, and robustness of the Pareto solutions. This study provides a robust framework for energy-efficient scheduling in complex manufacturing environments.

1. Introduction

Batch-processing machines (BPMs) are widely used in various industries, such as manufacturing, healthcare, and services, due to their ability to handle multiple jobs simultaneously, thus improving production efficiency and reducing operational costs (Fowler & Mönch, 2022). The scheduling of these machines is a critical task, especially when considering the dynamic nature of production environments and the necessity of preventive maintenance (PM) to ensure machine reliability. While PM plays a crucial role in extending the life of machinery and preventing unplanned breakdowns, it also introduces a significant challenge: downtime during maintenance directly affects machine availability, thereby impacting production efficiency. Consequently, production schedules must be carefully managed to balance the competing demands of maintaining machines and meeting production goals.

The parallel batch-processing machine scheduling problem with flexible preventive maintenance (UPBPM-FPM) is an NP-hard combinatorial optimization problem that has gathered attention in the field of industrial engineering due to its real-world relevance. In this problem, machines are required to process jobs in batches, with each machine capable of handling multiple jobs simultaneously. The objective is to minimize key performance metrics, such as makespan, earliness, tardiness, and total energy consumption, while simultaneously scheduling flexible preventive maintenance activities that ensure machines remain

* Corresponding author
E-mail mudassarrauf16@gmail.com (M. Rauf) lipeiwzu@163.com (P. Li)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)

doi: 10.5267/j.ijiec.2025.4.008

in optimal working condition. This problem becomes particularly complex when machines are “unrelated,” meaning each machine has distinct processing capabilities, energy consumption rates, and maintenance schedules.

Despite its significance in practical manufacturing systems, the UPBPM-FPM problem has not been extensively studied in the literature, particularly in the context of multi-objective optimization. To address this gap, this paper proposes a novel solution based on a Mixed-Integer Programming (MIP) model and a multi-objective artificial bee colony (MOABC) algorithm. The proposed algorithm incorporates three primary innovations that significantly enhance its ability to solve the UPBPM-FPM problem effectively:

Processing power-feature information (PP-FI) heuristic: This heuristic is designed to generate high-quality initial solutions by leveraging problem-specific characteristics, such as machine processing power and job features. By utilizing the processing power of machines and the specific characteristics of jobs, the PP-FI heuristic ensures that the initial solution is well-optimized, providing a strong starting point for further optimization.

Hybrid selection strategy: To improve the diversity and convergence of the algorithm, we introduce a hybrid selection strategy that combines the hypervolume index and the roulette wheel method. This approach strikes a balance between global exploration and local exploitation, ensuring that the algorithm explores a wide solution space while also focusing on refining solutions near the Pareto-optimal front.

Neighborhood search strategies: The MOABC algorithm employs both random and goal-oriented neighborhood search techniques to refine solutions and enhance the exploration and exploitation capabilities. These strategies allow the algorithm to navigate the solution space more effectively, ensuring that the Pareto frontier is well-distributed, and the objectives are appropriately balanced.

By combining these innovations, the MOABC algorithm is able to achieve superior performance in terms of convergence, diversity, and robustness when compared to traditional approaches, such as NSGA-III, ABC, and PSO. The experimental results presented in this study demonstrate that the MOABC algorithm outperforms these classical methods in solving the UPBPM-FPM problem, leading to more efficient scheduling solutions for complex production environments.
The remainder of this paper is organized as follows: Section 2 presents the literature review, Section 3 outlines the problem formulation and MIP model, Section 4 details the MOABC algorithm, Section 5 discusses experimental results, and Section 6 concludes with directions for future research.

2. Literature Review

2.1 BPM scheduling with preventive maintenance

BPM scheduling has been an area of extensive research, especially when coupled with PM. Several studies have explored various heuristics and metaheuristics to improve scheduling efficiency in the presence of maintenance activities. Huang et al. (2020) proposed a modified genetic algorithm to tackle the single BPM scheduling problem, incorporating flexible preventive maintenance to minimize downtime while maintaining production efficiency. Jang et al. (2022) developed a three-stage ant colony optimization (ACO) algorithm for parallel BPM scheduling that accounts for maintenance activities and job delays. Beldar et al. (2022) used simulated annealing (SA) and variable neighborhood search (VNS) approaches to minimize total tardiness in parallel BPMs, highlighting the importance of addressing maintenance while reducing delays. Real-world production systems often involve Unrelated Parallel Batch-Processing Machines (UPBPMs), where machines exhibit different processing times and capacities (Zeng & Liu, 2024). This adds complexity to scheduling problems, as it requires more sophisticated approaches. Jia et al. (2015) addressed the UPBPM scheduling problem by using a First-Fit-Decreasing heuristic along with the Max-Min Ant System (MMAS), accommodating the varying capacities of different machines. Suhaimi et al. (2016) used a Lagrangian relaxation approach for scheduling on UPBPMs, focusing on optimizing the overall makespan. Arroyo et al. (2019) introduced an iterative greedy algorithm for scheduling jobs of varying sizes and dynamic release times on UPBPMs, improving the flexibility of scheduling while reducing tardiness.

2.2 Multi-objective optimization of scheduling problem

While early research on BPM scheduling predominantly focused on single-objective optimization, there has been a growing shift toward multi-objective approaches that consider multiple conflicting objectives simultaneously. Minimizing makespan has traditionally been a primary focus, as seen in Xiao et al. (2024) , who introduced a tabu-based adaptive large neighborhood search algorithm for minimizing makespan on UPBPMs. Similarly, Zhang et al. (2020) applied a genetic algorithm combined with a heuristic placement strategy to achieve the same goal. Jia et al. (2020) proposed two heuristic algorithms and an ant colony optimization approach to minimize the total weighted delivery time on UPBPMs. However, modern manufacturing systems also face challenges related to energy consumption and environmental impact.

Wang et al. (2021) proposed a three-population co-evolutionary algorithm to solve a bi-objective problem involving makespan and total energy consumption on parallel BPMs, a common concern in the era of green manufacturing. Jia et al. (2017) introduced a Pareto-based ACO algorithm for minimizing both makespan and energy consumption on identical parallel BPMs, contributing to the multi-objective optimization field. Li et al. (2022) focused on discrete bi-objective evolutionary algorithms to minimize maximum lateness and pollution emission costs, showing the increasing importance of sustainability alongside traditional production efficiency. The incorporation of energy consumption into scheduling problems is particularly important in light of modern industrial practices, where reducing energy usage can lead to significant cost savings and environmental benefits. These studies laid the groundwork for the multi-objective scheduling problems addressed by this paper, which aims to simultaneously minimize makespan, tardiness, and energy consumption while incorporating flexible preventive maintenance. Metaheuristic algorithms, particularly population-based methods, have proven effective for solving complex optimization problems such as BPM scheduling. The Artificial Bee Colony (ABC) algorithm, introduced by Karaboga in 2005, is a well-established metaheuristic known for its simplicity and efficiency in handling multi-modal and multi-dimensional optimization problems. The ABC algorithm excels in maintaining solution diversity, preventing the search process from converging prematurely to suboptimal solutions. This feature is crucial when dealing with complex scheduling problems, where the solution space is vast and multi-objective. The ABC algorithm has been successfully applied to various scheduling and optimization problems, such as the one addressed in this paper. The algorithm requires fewer control parameters compared to other population-based methods, making it easier to implement and more reliable for industrial applications (Graham et al., 1979). Recent adaptations of the ABC algorithm have focused on improving solution quality by introducing local search enhancements, multi-objective extensions, and hybrid approaches (Graham et al., 1979). These modifications enhance the algorithm's performance in multi-objective scheduling, particularly in applications like the UPBPM-FPM problem, where multiple conflicting objectives must be balanced simultaneously.

Table 1
Comparison of the Proposed Method with Existing Batch Processing Machine Scheduling Methods

Ref.	Problem Characteristics				Maintenance	Objective	Algorithms
	Single	Identical	Uniform	Unrelated			
Huang et al. (2020)	√				√	C_{max}	Heuristics and Improved Genetic Algorithm
Jang et al. (2022)		√			√	C_{max}	Three-Stage ACO-Based Algorithm
Beldar et al. (2022)				√	√	total tardiness	Simulated Annealing and Variable Neighborhood Search
Zeng and Liu (2024)				√		C_{max}	Heuristic and Max-Min Ant System
Jia et al. (2015)				√		C_{max}	Lagrangian Relaxation approach
Suhaimi et al. (2016)				√		total flow time	Iterated Greedy Algorithm
Arroyo et al. (2019)				√		C_{max}	Tabu-based Adaptive Large Neighborhood Search
Xiao et al. (2024)				√		C_{max}	Genetic Algorithm
Zhang et al.				√		C_{max}	Full-batch longest processing time
Zhang et al. (2020)				√		total weighted delivery time	Heuristic and Ant Colony Optimization
Jia et al. (2020)				√		total service completion time	Heuristic and Improved Particle Swarm Optimization algorithm
Wang et al. (2021)				√		C_{max}, the total energy consumption	Three-populations Co-evolutionary Algorithm
Jia et al. (2017)		√				C_{max}, the total electric power cost	Pareto-based Ant Colony Optimization
Li et al. (2022)			√			maximum lateness, the total pollution emission costs	C-NSGA-A
This paper				√	√	C_{max}, earliness and tardiness, total energy consumption	MOABC

Table 1 summarizes the literature on BPM scheduling problems, highlighting their key characteristics, optimization objectives, and proposed algorithms. In summary, while BPM scheduling has advanced with a focus on single-objective optimization, recent research increasingly considers multi-objective approaches that balance traditional production metrics with energy consumption and maintenance. The integration of FPM adds complexity, reflecting real-world challenges in manufacturing. Metaheuristics such as genetic algorithms, and PSO have been widely used, but newer approaches like the MOABC algorithm offer enhanced performance in multi-objective scenarios. This paper contributes by introducing novel strategies to improve solution diversity, convergence, and robustness, addressing the complex trade-offs between makespan, tardiness, energy consumption, and maintenance.

3. Problem description

3.1 Problem formation

This study addresses a scheduling problem involving a set of jobs with varying processing times, sizes, and due dates on UPBPMs. Each machine has distinct capacity and unit energy consumption. The goal is to simultaneously minimize makespan (C_{max}), earliness and tardiness (ET), and total energy consumption (TEC). To maintain the machines' performance in the expected processing condition, flexible preventive maintenance (FPM) activities must be incorporated into the schedule. It is assumed that the continuous operation time of a machine cannot exceed a predefined maintenance threshold, and the maintenance duration is fixed. Using the three-field classification method (Graham et al., 1979), this problem can be denoted as $R_m|d_j, s_j, Q_i, FPM|(C_{max}, ET, TEC)$, where R_m represents the unrelated parallel batch-processing machines, d_j and s_j represents the due date and size of the job J_j, Q_i represents the capacity of machine M_i. The primary decisions in this problem include: (1) Assigning jobs to unrelated machines. (2) Grouping jobs assigned to each machine into batches. (3) Scheduling the start time of batches on each machine. (4) Determining the start times of maintenance activities on each machine. The following assumptions apply to this problem: (1) All jobs are available for processing at time zero. (2) The total size of the jobs in a batch must not exceed the capacity of the machine to which it is assigned. (3) The processing time of a batch is determined by the job with the longest processing time within that batch. (4) Once a batch is initiated, it must be completed without interruption.

3.2 Mixed-integer programming model

The symbols and variables used to formulate the mathematical model are summarized in Table 2.

Table 2
Symbols and variables.

Notations	Meanings
J_j	The jth job in a job set J, $j = 1,2,\dots,n$
M_i	The ith machine in machine set M, $i = 1,2,\dots,m$
$B_{i,b}$	the bth batch in M_i, $b = 1,2,\dots,n$
$p_{i,j}$	The processing time of J_j in M_i
d_j	The due date of J_j
s_j	The size of J_j
Q_i	The capacity of M_i
$ST_{i,b}$	The start time of $B_{i,b}$
$P_{i,b}$	The processing time of $B_{i,b}$
$CT_{i,b}$	The completion time of $B_{i,b}$
$A_{i,b}$	The cumulative processing time or age of M_i before processing $B_{i,b}$
P_i^p	The processing power of M_i
P_i^i	The idle power of M_i
P_i^m	The maintenance power of M_i
PE	Total processing energy consumption
IE	Total idle energy consumption
ME	Total maintenance energy consumption
ET_j	The earliness and tardiness of job J_j
UT	The maintenance threshold
t^m	The maintenance duration
Γ	A very large positive integer
C_{max}	Makespan
ET	Earliness and tardiness
TEC	Total energy consumption
$W_{i,b}$	Binary auxiliary variable, 1, if $B_{i,b}$ is not empty, otherwise 0
$X_{i,j,b}$	Binary decision variable, 1, if J_j is assigned to $B_{i,b}$, otherwise 0
$Y_{i,b}$	Binary decision variable, 1, if maintenance of M_i before processing $B_{i,b}$, otherwise 0

Building on Li's Li et al. (2022) model, the mixed-integer programming (MIP) model for the current problem is formulated as follows.

$$f_1 = min\ (C_{max}) \tag{1}$$

$$f_2 = min\ (ET) \tag{2}$$

$$f_3 = min\ (TEC) \tag{3}$$

$$C_{max} \geq CT_{i,b}, i = 1,2, \dots, m; b\ = 1,2, \dots, n \tag{4}$$

$$ET \geq \sum_{j=1}^{n} ET_j \tag{5}$$

$$ET_j \geq d_j - CT_{i,b} - \Gamma(1 - X_{i,j,b}), i = 1,2, \dots, m; j = 1,2, \dots, n; b = 1,2, \dots, n \tag{6}$$

$$ET_j \geq CT_{i,b} - d_j - \Gamma(1 - X_{i,j,b}), i = 1,2, \dots, m; j = 1,2, \dots, n; b = 1,2, \dots, n \tag{7}$$

$$TEC = PE + IE + ME \tag{8}$$

$$PE = \sum\nolimits_{i=1}^{m} \sum\nolimits_{b=1}^{n} P_{i,b} * P_i^p \tag{9}$$

$$IE = \sum\nolimits_{i=1}^{m} (ST_{i,1} + \sum\nolimits_{b=2}^{n} (ST_{i,b} - CT_{i.b-1} - t^m * Y_{i,b}) * P_i^i) \tag{10}$$

$$ME = \sum\nolimits_{i=1}^{m} \sum\nolimits_{b=1}^{n} Y_{i,b} * t^m * P_i^m \tag{11}$$

$$\sum_{i=1}^{n} \sum_{b=1}^{n} X_{i,j,b} = 1\ , j = 1,2, \dots, m \tag{12}$$

$$\sum\nolimits_{j=1}^{n} s_j X_{i,j,b} \leq Q_i, i = 1,2, \dots, m; b = 1,2, \dots, n \tag{13}$$

$$P_{i,b} \geq p_{i,j} X_{i,j,b}, i = 1,2, \dots, m; j = 1,2, \dots, n; b = 1,2, \dots, n \tag{14}$$

$$ST_{i,1} \geq 0, i = 1,2, \dots, m \tag{15}$$

$$ST_{i,b+1} \geq CT_{i,b} + t^m * Y_{i,b}, i = 1,2, \dots, m; b = 1,2, \dots, n-1 \tag{16}$$

$$CT_{i,b} \geq ST_{i,b} + P_{i,b}, i = 1,2, \dots, m; b = 1,2, \dots, n \tag{17}$$

$$Y_{i,1} = 0, i = 1,2, \dots, m \tag{18}$$

$$A_{i,1} = 0, i = 1,2, \dots, m \tag{19}$$

$$A_{i,b+1} \geq A_{i,b} + P_{i,b} - \Gamma * Y_{i,b+1}, i = 1,2, \dots, m; b = 1,2, \dots, n-1 \tag{20}$$

$$A_{i,b+1} \geq P_{i,b} - \Gamma(1 - Y_{i,b+1}), i = 1,2, \dots, m; b = 1,2, \dots, n-1 \tag{21}$$

$$A_{i,b} \leq UT, i = 1,2, \dots, m; b = 1,2, \dots, n \tag{22}$$

$$W_{i,b} = \begin{cases} 1, if \sum\limits_{j=1}^{n} X_{i,j,b} \geq 1 \\ 0, otherwise \end{cases}, i = 1,2, \dots, m; b = 1,2, \dots, n \tag{23}$$

$$W_{i,b} \geq W_{i,b+1}, i = 1,2, \dots, m; b = 1,2, \dots, n-1 \tag{24}$$

$$X_{i,j,b} \in \{0,1\}, i = 1,2, \dots, m; j = 1,2, \dots, n; b = 1,2, \dots, n \tag{25}$$

$$Y_{i,b} \in \{0,1\}, i = 1,2, \dots, m; b = 1,2, \dots, n \tag{26}$$

The three objectives to be minimized simultaneously are C_{max}, ET and TEC, as expressed in Eqs. (1-3), respectively. The calculations for C_{max}, ET and TEC are summarized in Eqs. (4-11), where Eqs. (67) define the earliness and tardiness of jobs, and Eqs. (9-11) specify the machines' processing, idle, and maintenance energy consumption, respectively. Constraint (12) ensures that each job is assigned to a batch and processed on a single machine. Constraint (13) guarantees that the total size of jobs in a batch does not exceed the machine's capacity. Constraint (14) defines the processing time of the batch. Constraints (15)-(16) determine the start time of the batch. Constraint (17) specifies the completion time of the batch. Constraint (18) ensures that maintenance will not be performed before processing the first batch on each machine. Constraints (19)-(21) define the cumulative processing time of the machine before starting the batch. Constraint (22) ensures that the cumulative processing time of the machine does not exceed the maintenance threshold. Constraint (23)-(24) ensures that the allocation of batches must follow a continuous sequence, and no empty batches are allowed in between. Finally, Constraints (25)-(26) represent the binary decision variables.

3.3 An illustrative instance

This section describes a specific, simple instance consisting of two machines and 10 jobs, which helps to illustrate the proposed problem. The parameters for this problem instance are defined as follows: $P_1^p = 12$, $P_2^p = 15$, $P_1^i = 4$, $P_2^i = 5$, $P_1^m = 132$, $P_2^m = 165$, $Q_1 = 12$, $Q_2 = 15$, $UT = 40$, $t^m = 5$. Additional details about the jobs are provided in Table 3.

Table 3
The details of the ten jobs

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
$p_{1,j}$	25	25	47	32	7	10	40	26	3	4
$p_{2,j}$	12	27	23	39	6	21	17	32	12	37
s_j	4	8	6	1	10	6	2	7	10	3
d_j	124	129	142	141	111	118	138	134	108	126

We solved the instance using CPLEX with the objective of minimizing the C_{max}. The Gantt chart of solution for this case is shown in Fig.1. The solution obtained from the CPLEX model consists of three batches on machine M_1 and two batches on machine M_2. In this instance, C_{max} is 50, ET is 958, and TEC is 2700.

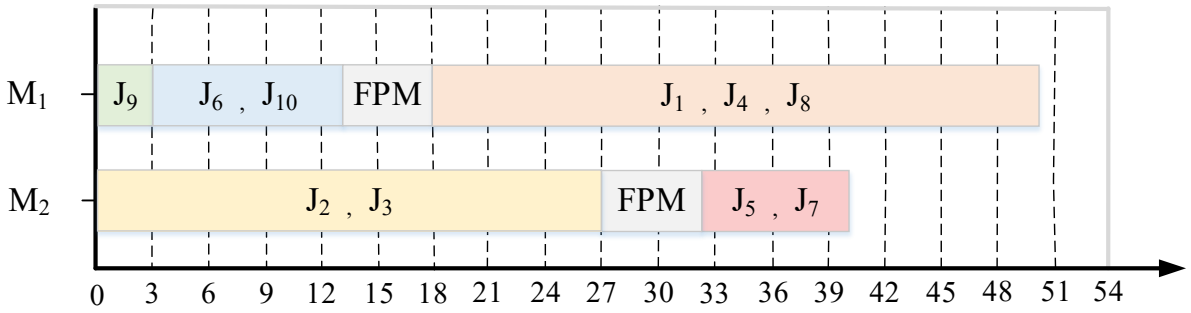


Fig. 1. Gantt chart of a solution for an example.

4. Multi-objective artificial bee colony algorithm

The UPBPM-FPM problem involves a large number of decision variables and a vast solution space, posing significant computational challenges. To address these issues, the MOABC algorithm was specifically designed to enhance the efficiency and effectiveness of the conventional ABC algorithm (Graham et al., 1979). The flowchart of the MOABC algorithm is shown in Fig. 2. This algorithm incorporates three key features to improve its performance: (1) PP-FI heuristic is introduced to generate a high-quality initial population by leveraging problem-specific characteristics. The integration of the FF heuristic ensures both the convergence and diversity of the initial solutions. (2) The algorithm incorporates a selection strategy combining hypervolume metrics with roulette wheel methods, striking a balance between global exploration and local exploitation to improve solution quality. (3) A diverse set of random and targeted neighborhood search operators enhances the algorithm's ability to refine solutions, ensuring a well-distributed Pareto frontier. These features collectively improve the computational efficiency, solution quality, and robustness of the MOABC algorithm, making it well-suited for solving the complex UPBPM-FPM problem.

4.1 Chromosome representation

A two-layer encoding method is proposed to represent the results of job allocation and batch formation. The first layer denotes the machine index, indicating the assigned machine for each job. The second layer represents the batch index, specifying the batch to which each job belongs on the corresponding machine. An example of this encoding method for a problem with ten

jobs and two UPBPMs is illustrated in Fig. 3. In this example, J_1, J_3, J_6 and J_8 are assigned to M_1, while the remaining jobs are assigned to M_2. on M_1, the first batch consists of J_1 and J_6, while the second batch includes J_3 and J_8. Similarly, on M_2 the first batch consists of J_4 and J_9, and the second batch contains J_2, J_5, J_7 and J_{10}.

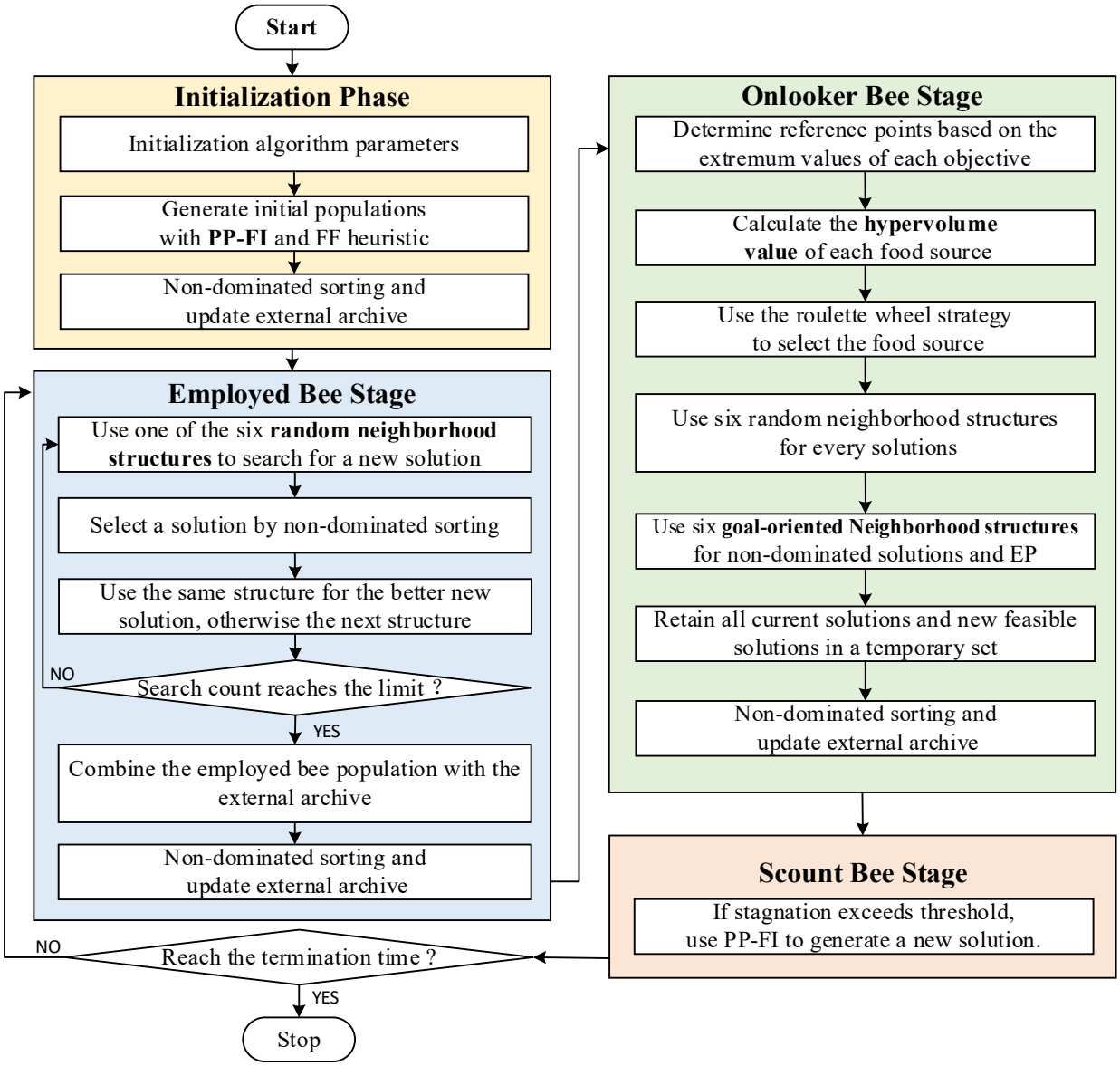


Fig. 2. Flowchart of the proposed MOABC Algorithm.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Machine(i)	1	2	1	2	2	1	2	1	2	2
Batch(b)	1	2	2	1	2	1	2	2	1	2

Fig. 3. Example of encoding.

4.2 Initial population generation methods

To ensure diversity and quality in the initial population, a portion P_p of the individuals is generated using the First-Fit (FF) heuristic (Graham et al., 1979), while the remaining individuals are created using the PP-FI heuristic. This hybrid strategy leverages the simplicity of the FF heuristic to ensure rapid generation of diverse initial solutions while integrating the PP-FI heuristic to enhance the overall solution quality.

4.2.1 PP-FI heuristic

To improve the quality of batch formation, the PP-FI heuristic begins by selecting a machine based on its processing power characteristic values using the roulette wheel method to initiate an empty batch. Subsequently, three job candidate sets

including the jobs that can be assigned to this batch are generated. A job is then selected from one of these sets using the roulette wheel method, and all job candidate sets are updated accordingly. This iterative process continues until the batch is fully formed. If any jobs remain unassigned to a batch, the procedure restarts from the beginning. The flow of this heuristic is illustrated in Fig. 4, ensures a balanced assignment of jobs to batches while leveraging machine processing power to optimize overall performance.

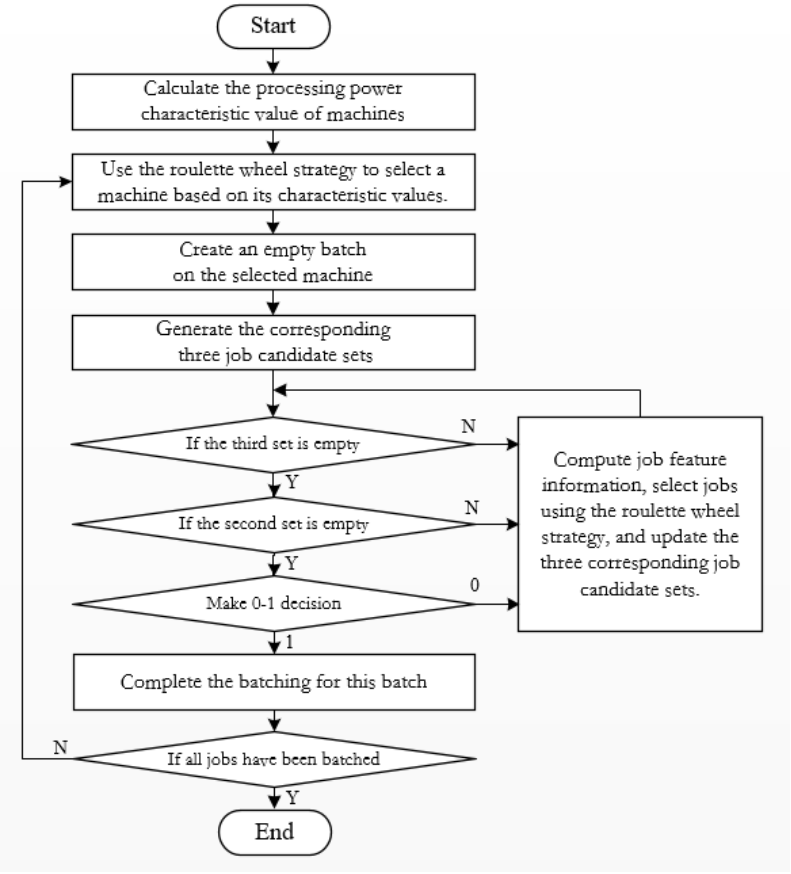


Fig. 4. The flow of PP-FI heuristic.

The processing power characteristic value (pp_i) of the machine M_i is calculated by Equation (23).

$$pp_i = \frac{\frac{1}{P_i^p}}{\sum_{i=1}^{m} \frac{1}{P_i^p}} \tag{23}$$

To efficiently allocate all feasible unassigned jobs to batch B_{i,b_i}, the candidate jobs are divided into three candidate sets: L_{i,b_i}^1, L_{i,b_i}^2, and L_{i,b_i}^3. The first set L_{i,b_i}^1 includes jobs whose sizes do not exceed the remaining capacity of the machine M_i in the current batch b_i, as defined in Eq. (24). The second set L_{i,b_i}^2 is derived from L_{i,b_i}^1 and consists of jobs whose cumulative processing time remains below the maintenance threshold, as shown in Eq. (25). Finally, the third set L_{i,b_i}^3 is a subset of L_{i,b_i}^2 and includes jobs that do not increase the batch processing time, as illustrated in Eq. (26). By prioritizing the inner sets, this approach ensures a balanced trade-off between machine capacity, maintenance constraints, and batch processing efficiency, facilitating a more effective allocation strategy.

$$L_{i,b_i}^1 = \{J_j | s_j \le (Q_i - \sum_{J_j \in B_{i,b_i}} s_j), J_j \in J\} \tag{24}$$

$$L_{i,b_i}^2 = \{J_j | max\{p_{i,j}, P_{i,b_i}\} + A_{i,b_i} \le UT, J_j \in L_{i,b_i}^1\} \tag{25}$$

$$L_{i,b_i}^3 = \{J_j | p_{i,j} \le P_{i,b_i}, J_j \in L_{i,b_i}^2\} \tag{26}$$

To address the multi-objective optimization nature of the problem studied in this paper, three key factors are considered when selecting a job from the candidate sets.

(1) Batch-processing machine characteristics: Jobs with processing times closest to the current batch-processing time are prioritized. This strategy enhances batch utilization and contributes to optimizing the total energy consumption (TEC) objective.

(2) Unrelated parallel machine characteristics: Priority is given to the job with the most significant time difference (TD) in processing time, defined as the difference between the minimum and second minimum processing times across machines. This approach reduces the total processing time and optimizes the C_{max} objective.

(3) Due date characteristics: Jobs with earlier due dates are prioritized over those with later due dates, ensuring better control over ET objective.

Combining these three factors, the calculation method for the job feature information (FI) value when a job J_j is assigned to batch B_{i,b_i} is formulated, as shown in Eq. (27). Two specific conditions are defined for special cases: (1) When batch B_{i,b_i} is empty, $\left|P_{i,b_i} - p_{i,j}\right|$ is set to zero; (2) When job J_j is not assigned to the machine corresponding to its minimum processing time, TD_j is set to zero. The percentage representation of the job feature information value is defined in Eq. (28).

$$fi_{i,j,b_i} = \frac{TD_j + 1}{(\left|P_{i,b_i} - p_{i,j}\right| + 1) * d_j}, J_j \in L_{i,b_i} \tag{27}$$

$$fi^*_{i,j,b_i} = \frac{fi_{i,j,b_i}}{\sum fi_{i,j,b_i}}, J_j \in L_{i,b_i} \tag{28}$$

When both set L^2_{i,b_i} and set L^3_{i,b_i} are empty, a 0-1 decision variable is introduced to determine whether to select a job from set L^1_{i,b_i}. This decision is designed to balance two key factors: the capacity utilization of the machine and the continuity of the batch's machining time. The procedure of the PP-FI heuristic is outlined in Algorithm 1.

Algorithm 1: *PP-FI heuristic*

Set UJS(set of unassigned jobs) $= J_j$, b_i(batch index of M_i) $= 0$,
    A_{i,b_i}(cumulative machining time of M_i prior to processing B_{i,b_i}) $= 0$
Calculate the pp_i of machine M_i according to Equation (23)
While $UJS \neq \emptyset$ **do**
  Select a machine M_i using the roulette wheel strategy according to the value of pp_i
  $Set\ b_i = b_i + 1$
  Update job candidate sets $L^1_{i,b_i}, L^2_{i,b_i}, L^3_{i,b_i}$according to Equations $(24) - (26)$
    While $L^3_{i,b_i} \neq \emptyset$ **do**
      Select J_j from L^3_{i,b_i} using the roulette wheel strategy based on fi^*_{i,j,b_i}
      Perform the UpdateCandidateSets operation, which includes removing J_j from UJS and updating all candidate sets
    End while
    While $L^2_{i,b_i} \neq \emptyset$ **do**
      Select J_j from L^2_{i,b_i} using the roulette wheel strategy based on fi^*_{i,j,b_i}
      UpdateCandidateSets
    End while
    While $L^1_{i,b_i} \neq \emptyset$ **and** random$\{0,1\} = 0$ **do**
      Select J_j from L^1_{i,b_i} using the roulette wheel strategy based on fi^*_{i,j,b_i}
      UpdateCandidateSets
    End while
    If $P_{i,b_i} + A_{i,b_i} >$ UT
      $A_{i,b_i+1} = 0$
    Else
      $A_{i,b_i+1} = A_{i,b_i} + P_{i,b_i}$
    End if
End while

4.2.2 An illustrative example problem of PP-FI heuristic

This section presents a small example to demonstrate the procedure of job allocation and batch formation using the PP-FI heuristic. In this example, ten jobs are allocated to two UPBPMs. The parameters of this problem instance are identical to those described in Section 3.1. A Gantt chart of one feasible solution is shown in Fig. 5.

The processing power characteristic values pp_1 and pp_2 of machine M_1 and M_2 are 44.44% and 55.56%, respectively. Using the roulette wheel strategy, machine M_1 is selected, and $b_1 = b_1 + 1 = 0 + 1 = 1$, an empty batch $B_{1,1}$ is created. The job

candidate sets are generated as $L_{1,1}^2 = L_{1,1}^1 = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\}$. Since there is no job in $B_{1,1}$, $L_{1,1}^3 = \emptyset$, a job is selected from $L_{1,1}^2$. The feature information percentage values of jobs are 10.16%, 9.77%, 8.88%, 8.94%, 11.36%, $10.68\%, 9.13\%, 9.41\%, 11.67\%, 10.00\%$, and job J_6 is selected. Updating the job candidate sets, $L_{1,1}^2 = L_{1,1}^1 = \{J_1, J_3, J_4, J_7, J_{10}\}$, $L_{1,1}^3 = \{J_{10}\}$. Selecting job J_{10} from $L_{1,1}^3$ and update job candidate sets $L_{1,1}^2 = L_{1,1}^1 = \{J_4, J_7\}$, $L_{1,1}^3 = \emptyset$. Selecting job J_4 from $L_{1,1}^2$, and update the job candidate sets, $L_{1,1}^2 = L_{1,1}^1 = \{J_7\}$, $L_{1,1}^3 = \emptyset$. Selecting job J_7 from $L_{1,1}^2$, and update the job candidate sets, $L_{1,1}^1 = \emptyset$. Therefore, $B_{1,1}$ consists of job J_6, J_{10}, J_4, J_7. Since $A_{1,1} + \max\{p_{1,6}, p_{1,10}, p_{1,4}, p_{1,7}\} \leq UT$, update $A_{1,2} = A_{1,1} + \max\{p_{1,6}, p_{1,10}, p_{1,4}, p_{1,7}\} = 0 + \{10, 4, 32, 48\} = 48$.

Since UJS $\neq \emptyset$, the machine M_1 is selected and $b_2 = b_2 + 1 = 0 + 1 = 1$, an empty batch $B_{2,1}$ is created. The job candidate sets are generated as $L_{2,1}^2 = L_{2,1}^1 = \{J_1, J_2, J_3, J_5, J_8, J_9\}$. Since there is no job in $B_{1,1}$, $L_{1,1}^3 = \emptyset$, a job is selected from $L_{2,1}^2$. Since the feature information percentage value of jobs are $16.60\%, 15.59\%, 14.49\%, 18.54\%, 15.36\%, 19.06\%$, job J_5 is selected from $L_{2,1}^2$.Updating the job candidate sets, $L_{2,1}^2 = L_{2,1}^1 = \{J_7\}$, $L_{1,1}^3 = \emptyset$. Selecting job J_7 from $L_{2,1}^2$, and update the job candidate sets, $L_{2,1}^1 = \emptyset$. Therefore, $B_{2,1}$ consists of job J_5, J_1. Since $A_{2,1} + \max\{p_{2,5}, p_{2,1}\} \leq UT$, update $A_{2,2} = A_{2,1} + \max\{p_{2,5}, p_{2,1}\} = 0 + \{12\} = 12$.

Since $UJS \neq \emptyset$, the machine M_2 is selected and $b_2 = b_2 + 1 = 1 + 1 = 2$, an empty batch $B_{2,2}$ is created. The job candidate sets are generated as $L_{2,2}^2 = L_{2,2}^1 = \{J_1, J_2, J_3, J_8, J_9\}$, $L_{2,1}^3 = \emptyset$. Since the feature information percentage value of jobs are $20.37\%, 19.59\%, 17.79\%, 18.85\%, 23.39\%$, the job J_2 is selected. Updating the job candidate sets, $L_{2,2}^1 = \{J_3, J_8\}$, $L_{2,2}^3 = L_{2,2}^2 = \{J_3\}$, the job J_3 is selected. Updating the job candidate sets, $L_{2,2}^1 = \emptyset$. Therefore, $B_{2,2}$ consists of job J_2, J_3. Since $A_{2,2} + \max\{p_{2,2}, p_{2,3}\} \leq UT$, update $A_{2,3} = A_{2,2} + \max\{p_{2,2}, p_{2,3}\} = 12 + \{27, 23\} = 39$.

Since $UJS \neq \emptyset$, the machine M_1 is selected and $b_1 = b_1 + 1 = 1 + 1 = 2$, an empty batch $B_{1,2}$ is created. The job candidate sets are generated as $L_{1,2}^2 = L_{1,2}^1 = \{J_8, J_9\}$, $L_{1,3}^3 = \emptyset$, a job is selected from $L_{1,2}^2$. The feature information percentage values of jobs are $44.63\%, 55.37\%$, and the job J_8 is selected. After processing $B_{1,2}$ the accumulated processing time of the machine will exceed UT. Therefore, $A_{1,2} = 0$. Updating the job candidate sets, $L_{1,2}^1 = \emptyset$. Therefore, and $B_{1,2}$ consists of J_8. Since $A_{1,2} + \max\{p_{1,8}\} \leq UT$, update $A_{1,3} = 0 + \max\{p_{1,8}\} = 0 + \{26\} = 26$.

Since $UJS \neq \emptyset$, the machine M_1 is selected and $b_1 = b_1 + 1 = 2 + 1 = 3$, an empty batch $B_{1,3}$ is created. The job candidate sets are generated as $L_{1,3}^2 = L_{1,3}^1 = \{J_9\}$, $L_{1,3}^3 = \emptyset$, jobJ_9 is selected from $L_{1,3}^2$. Updating the job candidate sets, $L_{1,3}^1 = \emptyset$. Therefore, and $B_{1,3}$ consists of J_9. Since $A_{1,3} + \max\{p_{1,9}\} \leq UT$, update $A_{1,4} = A_{1,3} + \max\{p_{1,9}\} = 26 + \{3\} = 29$.

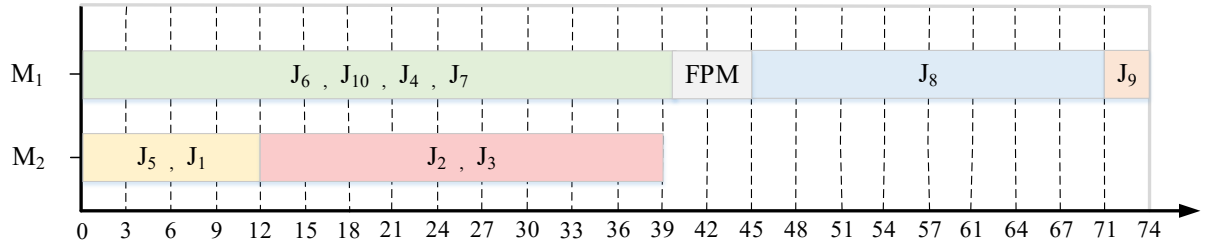


Fig. 5. Instance solution based on PP-FI heuristic.

Following the PP-FI procedure, the solution obtained is with three batches in the machine M_1 and two batches in the machine M_2. In this instance, C_{max} is 74, ET is 864, and TEC is 2073.

4.3 Decoding with RLS rule

Since the objective of the ET is related to the start time of batch and maintenance, after determining the job allocation and batch formation, a Right-Left Shifting (RLS) rule is proposed to adjust the start times of batches and maintenance activities. Equation (28) defines the median due date $\tilde{d}$ of jobs in batch $B_{i,b}$, where $n_{i,b}$ represents the number of jobs in batch $B_{i,b}$.

$$\tilde{d} = \begin{cases} d_{(n_{i,b}+1)/2}, & if\ n_{i,b}\ is\ odd \\ d_{n_{i,b}/2}, & if\ n_{i,b}\ is\ even \end{cases} \tag{28}$$

Since the batch completion time $CT_{i,b}$ approaches $\tilde{d}$, the batch's ET value decreases (Zhang et al., 2021). When $CT_{i,b} \leq \tilde{d}$, $B_{i,b}$ is shifted to the right, to further reduce the ET value among different batches. To formalize this, we define a $Block$ as a sequence of continuous processing batches (including maintenance) without idle time (Zhang et al., 2022). Let $|A_p^{\mathrm{t}}|$, $|A_p^{\mathrm{i}}|$ and $|A_p^{\mathrm{e}}|$ denote the number of tardy jobs, punctual jobs, and early finished jobs in $Block\ p$, respectively. Define $\Delta_p = |A_p^{\mathrm{t}}| - |A_p^{\mathrm{i}}| - |A_p^{\mathrm{e}}|$, where $\Delta_p \geq 0$ indicates that the number of punctual and early jobs exceeds the number of tardy jobs. When

$CT_{i,b} > \tilde{d}$ and $\Delta_p \geq 0$, shifting $Block_p$ to the left can potentially yield a new solution with a smaller makespan compared to the current solution. Although shifting a block left or right moves $CT_{i,b}$ closer to $\tilde{d}$ and reduces the ET value, it may also increase the C_{max} and the TEC. To address this trade-off, a binary variable is randomly generated to determine whether the shift will occur. Additionally, If the cumulative processing time of the machine after processing the batch exceeds the maintenance threshold, maintenance will be scheduled immediately after the previous batch is completed. The detailed procedure for decoding based on the RLS rule is presented in Algorithm 2.

Algorithm 2: *Decoding based on RLS rule*

Input: n_i^b(the number of batches in the machine M_i)
Output: $ST_{i,b}$(the start time of $B_{i,b}$), C_{max}, ET and TEC values of the solution

Set $i(\text{machine index}) = 1, b(\text{batch index}) = 1$
For $i = 1\ to\ m$ **then**
  For $b = 1\ to\ n_i^b$ **then**
    If $A_{i,b} + P_{i,b} > UT$ **then**
      $Y_{i,b} = 1, A_{i,b} = 0$
    Else
      $Y_{i,b} = 0, A_{i,b} = A_{i,b-1} + P_{i,b}$
    End if
    If $\text{random}\{0,1\} = 1$ **then**
      If $CT_{i,b} \leq \tilde{d}$ **then**
        $ST_{i,b} = \tilde{d} - P_{i,b}$
      While $\Delta_p \geq 0$ **and** $ST_{i,Block_p} \geq CT_{i,Block_{p-1}} + 1$ **then**
        $ST_{i,Block_p} = ST_{i,Block_p} - 1$, **Update** $Block_p$, $|A_p^t|$, $|A_p^i|$, $|A_p^e|$
        If $ST_{i,Block_p} = CT_{i,Block_{p-1}}$ **then**
          Update $Block_p$, $|A_p^t|$, $|A_p^i|$, $|A_p^e|$
          Break
        End if
      End while
    Else
      $ST_{i,b} = CT_{i,b-1} + Y_{i,b-1} * t^m$
    End if
  End for
End for
Calculate the values of C_{max}, ET, TEC

4.4 Employed Bee Phase

During the employed bee phase, each employed bee utilizes a neighborhood structure to search for new food sources. A total of six types of neighborhood structures are designed, balancing the exploration and exploitation capabilities of the algorithm. Sequential neighborhood (SN) searches are performed SN times for each employed bee. These neighborhood structures are described as follows, with a detailed demonstration provided in Fig. 6. The procedure for the employed bee phase is outlined in Algorithm 3.

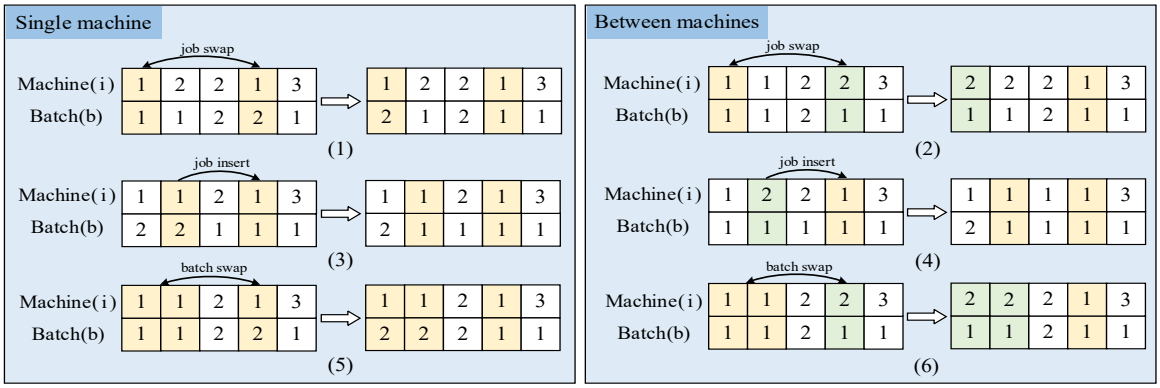


Fig. 6. Example of neighborhood structure.

(1) Single-machine job swap: Select one job randomly from each of two batches on the same machine and swap them.

(2) Between machine job swap: Select one job randomly from each batch on two different machines and swap them.
(3) Single-machine job insert: Select one job randomly from a batch on a machine and insert it into another batch on the same machine.
(4) Between-machine job insert: Select one job randomly from one machine and insert it into a random batch on another machine.
(5) Single-machine batch swap: Select two batches randomly from a machine and swap them.
(6) Between machine batch swap: Select one batch randomly from each machine and swap them.

Algorithm 3: *Employed Bee Phase*

Input: $\bar{P}$(initial population), NP(population size), SN(number of searches)
Output: EBP (employed bee population)
Set p(individual index) $= 1$, l(search index) $= 1, k$(structure index) $= 1$
For $p = 1\ to\ NP$ **then**
  For $l = 1\ to\ SN$ **then**
    $\pi_{new}^{p} \leftarrow$Apply the kth neighborhood structure to generate a new feasible solution
    If $\pi_{new}^{p} \prec \pi_{old}^{p}$ (π_{new}^{p} dominate π_{old}^{p})**then**
      $\pi^{p} \leftarrow \pi_{new}^{p}$(update the current solution)
    Else if $\pi_{old}^{p} \prec \pi_{new}^{p}$ **then**
      $\pi^{p} \leftarrow \pi_{old}^{p}, k = k + 1$
    Else
      $\pi^{p} \leftarrow random\{\pi_{new}^{p}, \pi_{old}^{p}\}, k = k + 1$
    End if
    If $k > 6$ **then**
      $k = 1$
  End for
  $l = 1$
End for

4.5 Onlooker Bee Phase

During the scout bee phase, additional exploitation is performed based on the optimal solutions identified in the employed bee phase. To facilitate this, the objective value of the food source from the employed bee phase are normalized using Equation (29). Here, $f_r(x)$ denotes the rth objective value of solution x, where $r = 1,2,3$. The minimum and maximum objective values across all individuals in the population are represented as $z_r^{min} = \min f_r(x)$ and $z_r^{max} = \max f_r(x)$, respectively, with X denoting the entire population. A reference point z_r^*, utilized for hypervolume value, is defined in Equation (30). The hypervolume percentage of each individual is then determined using Equation (31), which provides a measure of solution quality. Food source selection during this phase integrates hypervolume values with a roulette wheel strategy to ensure a balance between exploration and exploitation.

$$f_r'(x) = \frac{f_r(x) - z_r^{min}}{z_r^{max} - z_r^{min}}, \mathrm{x} \in \mathrm{X} \tag{29}$$

$$z_r^* = f_r^{max} + \sigma(f_r^{max} - f_r^{min}), \sigma \in (0,1) \tag{30}$$

$$hv_p^* = \frac{hv_p}{\sum_{p=1}^{NP} hv_p} \tag{31}$$

To address the limited capacity, which can lead to local optima (Li et al., 2022), six goal-oriented neighborhoods are designed to search for non-dominated solutions, ensuring solution are as close as possible to the Pareto frontier. All feasible solutions are stored in a temporary set, followed by non-dominated sorting and updating the external pareto archive (EPA). The key terms are defined as follows: CM_{max} represents the machine with the longest completion time, CM_{min} denotes the machine with the shortest completion time, TM_{max} refers to the machine with the highest processing energy consumption, and TM_{min} denotes the one with the lowest processing energy consumption. The six goal-oriented neighborhoods are detailed below.

(1) C_{max} job search: Swap the job with the largest $p_{i,j}$ in each batch on CM_{max} with each job on CM_{min}, retaining all feasible solution.
(2) C_{max} batch search: Swap each batch on CM_{max} with each batch on CM_{min}, retaining all feasible solutions.
(3) TEC job search: Swap the job with the largest $p_{i,j}$ in each batch on TM_{max} with each job on TM_{min}, retaining all feasible solution.
(4) TEC batch search: Swap each batch on TM_{max} with each batch on TM_{min}, retaining all feasible solutions.
(5) ET job search: Swap the job with the largest ET value with each job within the remaining batch on the same machine, retaining all feasible solutions.
(6) ET batch search: Swap the batch with the largest ET value with all other batches, retaining all feasible solutions.

These neighborhood strategies are designed to effectively balance the trade-offs between the objectives, including C_{max}, TEC, and ET. The procedure for the onlooker bee phase is outlined in Algorithm 4, which ensures thorough exploration of the solution space while maintaining a balance between exploration and exploitation.

Algorithm 4: *Onlooker Bee Phase*

Input: EBP (employed bee population), NP(population size), SN(number of searches)
Output: OBP(onlooker bee population)
Set p(individual index) $= 1$, l(search index) $= 1$, k(structure index) $= 1$,the feasible solution set $TS = \emptyset$
For $p = 1\ to\ NP$ **then**
  For $l = 1\ to\ SN$ **then**
    π_{new}^{p} ←Apply the kth neighborhood structure to generate a new feasible solution
    If $\pi_{new}^{p} < \pi_{old}^{p}$ **then**
      $\pi^{p} \leftarrow \pi_{new}^{p}$
    Else if $\pi_{old}^{p} < \pi_{new}^{p}$ **then**
      $\pi^{p} \leftarrow \pi_{old}^{p}, k = k + 1$
    Else
      $\pi^{p} \leftarrow random\{\pi_{new}^{p}, \pi_{old}^{p}\}, k = k + 1$
    End if
    If $k > 6$ **then**
      $k = 1$
  End for
  $l = 1$
End for
Set $p = 1$, $k = 1$
EBP^{*} ← non-dominated sorting EBP and retain non-dominated solutions
For $p = 1\ to\ |EBP^{*} \cup EPA|$ **then**
  For $k = 1\ to\ 6$ **then**
    $TS \leftarrow EBP^{*} \cup EPA$ use kth goal-oriented neighborhood until a feasible solution
  End for
End for
Non-dominated sorting $\{TS \cup EBP \cup EPA\}$
Update OBP and EPA

4.6 Scout Bee Phase

The scout bee phase plays a critical role in maintaining population diversity and preventing premature convergence. During this phase, the algorithm monitors the number of iterations for which each food source remains stagnant, i.e., without being updated. If a food source remains unchanged for a specified number of iterations, known as the limit, it indicates that the source is no longer yielding new or beneficial solutions. To mitigate this issue, the algorithm replaces stagnant food sources with new ones generated using the PP-FI heuristic. This ensures that the search process is reinvigorated by introducing new candidate solutions into the population. By regenerating stagnant food sources, the scout bee phase facilitates the exploration of previously unvisited areas in the search space. This not only enhances the algorithm's global search capability but also increases the likelihood of escaping local optimum and finding the global optimum, thereby improving the overall robustness and performance of the algorithm.

5. Experimental design and result analysis

5.1 Experimental design

Based on actual production data from an enterprise (Li et al., 2022), we categorized the test problem instances into three groups: small, medium, and large. The categories are defined as follows: Small size(n={10,15,20}, m=2, UT=60, t^{m}=10), Medium size (n={40,60,80}, m={2,3}, UT=60, t^{m}=10), and Large size (n={100,150,200,250}, m={3,5}, UT=120, t^{m}=20). Each parameter combination generated ten instances, resulting in a total of 170 instances. The processing times ($p_{i,j}$) were sampled from a uniform distribution U [1,50], the sizes (s_j) from U [1,10], and the due dates (d_j) were calculated using Eq. (32) and Eq. (33) with τ=0.5 and ρ=0.2.

$$d_j = \sum_{i=1}^{m} \frac{p_{i,j}}{m} + U\left[d_{min}, d_{min} + \rho * \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{p_{i,j}}{m}\right]/2 \tag{32}$$

$$d_{min} = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{p_{i,j}}{m} * (\tau - \rho/2) \tag{33}$$

To ensure fairness in experimental comparisons, the maximum running time (T_{max}) was used as the termination condition (Jia et al., 2017), where $T_{max} = m * n * 360$ms. Each instance of all algorithms was executed independently ten times. The population size of all comparison algorithms was kept identical to that of the MOABC algorithm, and the crossover and mutation probabilities of NSGA-III were set to 0.9 and 0.1, respectively (Yuan et al., 2014). Four metrics were used to assess the performance of the multi-objective optimization algorithm (Li & Yao, 2019). (1) Generational Distance (GD) measures the algorithm's convergence. (2) Inverse Generational Distance (IGD) evaluates both uniformity and convergence comprehensively. (3) Non-dominance Rate (NR) assesses the diversity and convergence of the solutions. (4) C (A, *) represents the average of the C matrix for Algorithm A compared to other algorithms and measures the divergence between two Pareto frontiers.

5.2 Parameters tuning

The performance of the MOABC algorithm is governed by four key parameters: population size (NP), the number of neighborhood searches per individual (SN), the ratio of initial solutions generated by the FF heuristic (P_p), and the parameter σ for the reference point. Preliminary experiments identified four levels for each parameter, as summarized in Table 4. To streamline the parameter tuning process, a medium-sized representative problem was selected (Rauf et al., 2020; Yue et al., 2019). The Taguchi experiment design was employed to determine the optimal parameter combinations. Using Minitab, C_{max}, ET and TEC were set as response variables to compute the signal-to-noise (S/N) ratio. This approach resulted in $L_{16}(4^4)$ parameter combinations. Each experimental was evaluated through ten independent runs, leading to a total of 160 experiments.

Table 4
Algorithm parameters and values.

Parameters	Value			
	1	2	3	4
NP	50	100	150	200
SN	2	3	4	5
P_p	0.3	0.4	0.5	0.6
σ	0.2	0.4	0.6	0.8

The mean S/N ratio values were analyzed to identify the optimal parameter settings by illustrating the impact of each parameter through S/N plots, as shown in Fig. 7. Since this is a minimization problem, the 'larger-the-better' criterion was adopted for the S/N ratio in this study. For the multi-objective optimization problem, the mean S/N values were normalized to determine the optimal parameter settings by integrating the three objectives. This study employed the data normalization method proposed by Yue et al. (2019). Based on the normalized S/N ratio values for each objective, the derived optimal parameter settings for the algorithm are $NP = 150$, $SN = 2$, $P_p = 0.7$ and $\sigma = 0.8$.

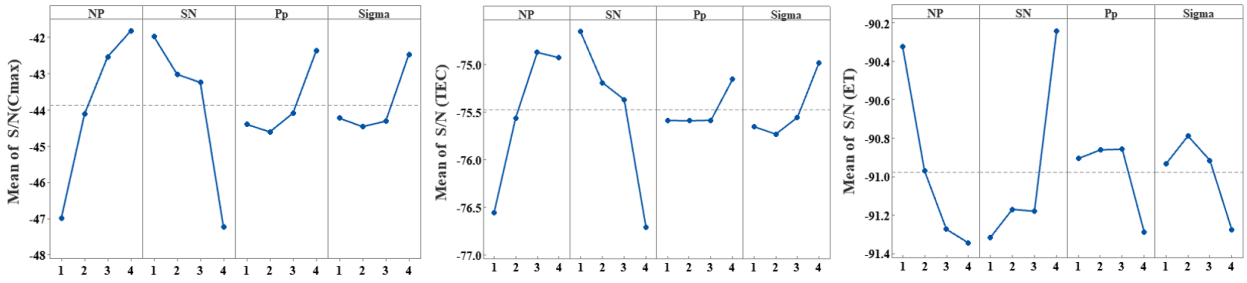


Fig. 7. Main effects of key parameters of MOABC.

5.3 Results and analysis

5.3.1 Performance comparison

Fig. 8 illustrates the convergence performance of the proposed MODABC algorithm in comparison with NSGA-II, ABC, and PSO across three objectives—C_{max}, TEC, and ET—under a large-scale problem instance.

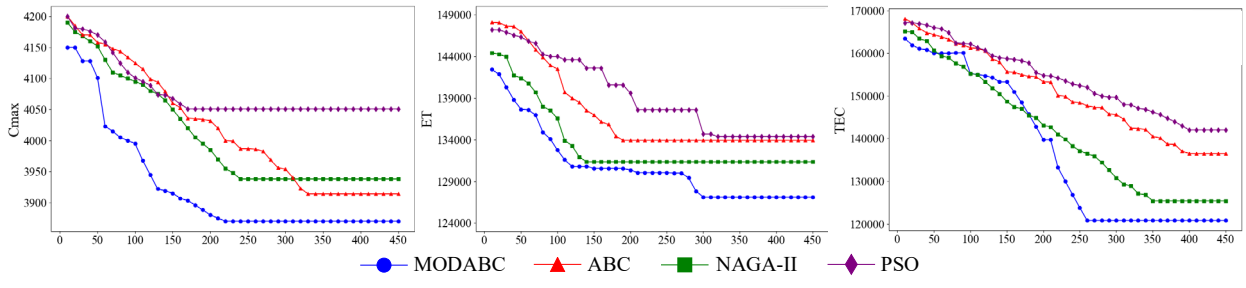


Fig. 8. Large-Scale Algorithm Convergence Comparison.

For the C_{max} objective, MODABC demonstrates significant convergence advantages, achieving the fastest convergence speed and the best final solution. ABC and NSGA-II exhibit relatively similar performance, with stable convergence speeds and final solution quality, though slightly inferior to MODABC. PSO performs the worst, with slow convergence speed, prolonged stagnation phases, and an inability to effectively escape local optima.

For the TEC objective, MODABC again shows outstanding performance, with a stable convergence curve and significantly better final solution quality compared to the other algorithms. NSGA-II follows, displaying a good convergence trend in the mid-stage but slowing down in the later stages. ABC and PSO exhibit weaker local search performance in this objective, leading to final solution quality notably inferior to MODABC and NSGA-II.

For the ET objective, MODABC maintains rapid and stable convergence characteristics, achieving the best final solution quality. NSGA-II converges quickly in the early stages but experiences significant fluctuations later, failing to reach an optimal solution. ABC and PSO perform the worst, with slower convergence speeds and significant fluctuations, failing to achieve satisfactory solutions.

Overall, MODABC effectively enhances global search and local optimization performance through its hypervolume contribution mechanism and multi-neighborhood search strategy, resulting in fast convergence and superior solution quality. NSGA-II performs relatively stable in multi-objective optimization but has limitations in local search precision. ABC and PSO struggle to balance global and local searches effectively, making it difficult to maintain a well-distributed Pareto front.

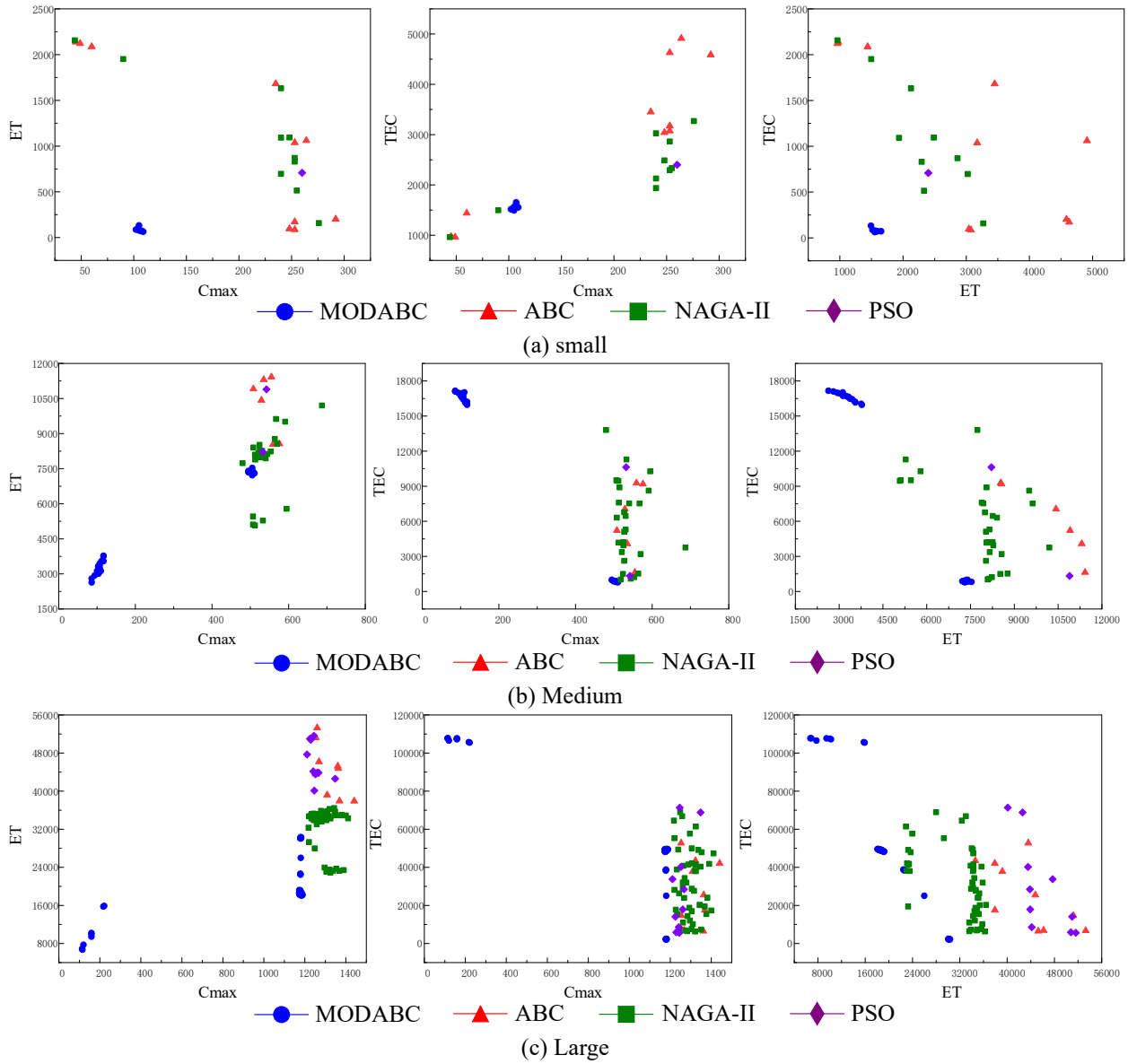


Fig. 9. Two dimensional Pareto front plot at three scales.

Fig. 9 presents the Pareto front results of the four algorithms, highlighting their performance differences in multi-objective optimization. As shown in Fig. 9(a), MODABC demonstrates balanced and superior performance across multiple objectives.

This can be attributed primarily to its use of the PP-FI heuristic method, which generates high-quality initial solutions by leveraging problem-specific characteristics, providing a solid foundation for optimization. Additionally, its neighborhood search strategy further enhances the diversity and balance of solutions, enabling better trade-offs among multiple objectives. In contrast, ABC and NSGA-II show slightly inferior performance in terms of the makespan objective, likely due to the lack of efficient initialization strategies, resulting in insufficient coverage of the solution space. PSO's Pareto front includes only a single solution, reflecting its limited exploration ability in complex multi-objective problems. As shown in Fig. 9(b) and Fig. 9(c), with increasing problem scale, MODABC continues to exhibit significant advantages in optimizing Cmax and ET objectives. This indicates that its robust local search capability enables it to maintain superior performance in complex problems. In comparison, ABC and PSO, lacking neighborhood strategies, are more prone to falling into local optima, limiting performance improvement. Although NSGA-II maintains good Pareto distribution in larger-scale problems, it fails to effectively incorporate problem-specific characteristics, resulting in slightly inferior performance in certain objectives compared to MODABC. Table 5 presents the analysis of the GD, IGD, and NR metrics for each instance, using two heuristic rules to generate feasible solutions. The feasible solutions are merged, and the non-dominated solutions are extracted to form the reference set. Furthermore, a C-matrix metric analysis is conducted on the non-dominated solutions within each solution set.

Table 5
The results obtained from initial solutions generated by PP-FI and FF heuristics.

Size	m-n	GD		IGD		NR		C-matrix	
		PP-FI	FF	PP-FI	FF	PP-FI	FF	C (PP-FI, FF)	C (FF, PP-FI)
Small	10-2	**21.15**	66.62	177.88	**64.92**	**0.88**	0.64	**0.35**	0.12
	15-2	**77.51**	107.09	**150.57**	231.18	**0.67**	0.61	**0.39**	0.33
	20-2	58.60	**18.61**	**190.49**	699.07	**0.85**	0.84	**0.15**	0.14
	Avg	**52.42**	64.11	**172.98**	331.72	**0.80**	0.70	**0.30**	0.20
Medium	40-2	**54.66**	1059.14	**632.84**	3179.98	**0.95**	0.3125	**0.68**	0.05
	40-3	**0.97**	781.12	**207.53**	1311.04	**0.97**	0.23	**0.76**	0.02
	60-2	**0.0**	6645.66	**0.0**	7203.36	**1.0**	0.0	**1.0**	0.0
	60-3	**0.0**	4473.72	**0.0**	4695.60	**1.0**	0.0	**1.0**	0.0
	80-2	**0.0**	**0.0**	**2060.27**	12568.34	**1.0**	**1.0**	**0.0**	**0.0**
	80-3	**0.0**	6351.85	**386.21**	8956.25	**1.0**	0.04	**0.95**	0.0
	Avg	**9.27**	3218.58	**547.81**	6319.10	**0.99**	0.26	**0.73**	0.01
Large	100-3	**0.0**	17680.82	**0.0**	19998.25	**1.0**	0.0	**1.0**	0.0
	100-5	**0.0**	9250.61	**0.0**	13380.47	**1.0**	0.0	**1.0**	0.0
	150-3	**0.0**	23095.34	**2390.17**	33304.42	**1.0**	0.04	**0.96**	0.0
	150-5	**0.0**	24657.64	**0.0**	25203.23	**1.0**	0.0	**1.0**	0.0
	200-3	**0.0**	30584.18	**0.0**	29661.18	**1.0**	0.0	**1.0**	0.0
	200-5	**0.0**	31805.23	**0.0**	39448.97	**1.0**	0.0	**1.0**	0.0
	250-3	**0.0**	37917.15	**0.0**	33094.72	**1.0**	0.0	**1.0**	0.0
	250-5	**0.0**	50580.17	**0.0**	57708.33	**1.0**	0.0	**1.00**	0.0
	Avg	**0.00**	28196.39	**298.77**	31474.95	**1.00**	0.01	**1.00**	0.00

The analysis in Table 5 reveals that the PP-FI heuristic performs marginally better than the FF heuristic in small-scale instances and significantly outperforms FF in medium and large-scale instances.

Table 6
The GD, IGD, and NR values were obtained from different algorithms.

Size	m-n	GD				IGD				NR			
		MOABC	ABC	NSGA-III	PSO	MOABC	ABC	NSGA-III	PSO	MOABC	ABC	NSGAIII	PSO
Small	10-2	**6.67**	629.44	445.53	1738.34	**629.44**	1784.35	1322.98	2106.82	**0.82**	0.10	0.07	0.01
	15-2	**3.60**	505.29	143.08	768.23	**505.29**	1354.32	324.47	1014.76	**0.50**	0.12	0.34	0.04
	20-2	**11.78**	424.90	215.39	476.68	**424.90**	1162.94	553.23	885.71	**0.43**	0.14	0.27	0.17
	Avg	**7.35**	519.88	268.00	994.42	**519.88**	1433.87	733.56	1335.76	**0.58**	0.12	0.23	0.07
Medium	40-2	**5.05**	1729.26	378.49	1231.59	**1729.26**	4783.46	1733.06	3031.94	**0.84**	0.00	0.13	0.03
	40-3	**29.50**	528.30	283.01	276.25	**528.30**	1704.19	666.53	1111.93	**0.45**	0.07	0.22	0.26
	60-2	**12.04**	2491.86	461.15	2626.23	**2491.86**	1162.94	553.23	885.71	**0.76**	0.01	0.23	0.01
	60-3	**24.19**	1393.27	577.97	1301.74	**1393.27**	4871.86	2611.09	3292.84	**0.63**	0.03	0.22	0.12
	80-2	**19.78**	5216.57	1057.99	4968.26	**5216.57**	12576.78	4368.47	9586.16	**0.75**	0.00	0.25	0.00
	80-3	**42.94**	2855.52	1029.75	3443.14	**2855.52**	12932.15	5473.46	12254.46	**0.43**	0.14	0.27	0.17
	Avg	**22.25**	2369.13	631.39	2307.87	**2369.13**	6338.56	2567.64	5027.17	**0.64**	0.04	0.22	0.10
Large	100-3	**59.45**	5829.50	1428.42	7589.56	**5829.50**	18753.57	8928.27	19737.29	**0.92**	0.00	0.07	0.01
	100-5	**81.94**	3861.72	1912.58	4592.01	**3861.72**	15290.35	11764.45	12123.63	**0.86**	0.02	0.10	0.02
	150-3	**62.10**	8830.36	1635.60	10638.25	**8830.36**	26087.09	9337.32	25203.75	**0.85**	0.00	0.15	0.00
	150-5	**76.66**	9128.86	3861.18	11693.33	**9128.86**	33992.52	24270.09	30958.68	**0.94**	0.01	0.04	0.01
	200-3	**163.63**	15747.64	1894.13	20141.38	**15747.64**	36880.52	9467.52	39446.83	**0.81**	0.00	0.19	0.00
	200-5	**258.19**	15206.15	5566.02	16444.94	**15206.15**	50372.80	33958.67	42455.79	**0.93**	0.01	0.05	0.00
	250-3	**136.95**	21063.12	5965.05	25315.85	**21063.12**	50205.06	28630.71	55285.08	**0.83**	0.00	0.17	0.00
	250-5	**136.05**	21763.39	5619.89	20396.98	**21763.39**	69113.32	31575.18	58492.98	**0.90**	0.00	0.09	0.00
	Avg	**121.87**	12678.84	3485.36	14601.54	**12678.84**	37586.90	19741.53	35463	**0.88**	0.01	0.11	0.01

This performance advantage can be attributed to the design of PP-FI, which considers batch utilization, variations in processing times for the same workpiece across different machines, and delivery deadlines. These features enable PP-FI to generate high-quality initial solutions for multi-objective optimization problems, particularly as the problem scale increases. Table 6 and Table 7 present the performance comparison of the MOABC algorithm against other algorithms across all instances. The best values for GD, IGD, NR, and the C-matric are highlighted in bold for each case, providing a clear assessment of algorithmic effectiveness.

Table 7
The values of the C-matric obtained by different algorithms.

Size	m-n	C (MOABC, ABC)	C (ABC, MOABC)	C (MOABC, NSGA-III)	C (NSGA-III, MOABC)	C (MOD ABC, PSO)	C (PSO, MOABC)	C (ABC, NSGA-III)	C (NSGA-III, ABC)	C (ABC, PSO)	C (PSO, ABC)	C (NSGA-III, PSO)	C (PSO, NSGA-III)
Small	10-2	**0.91**	0.00	**0.94**	0.00	**0.95**	0.00	0.34	**0.49**	**0.37**	0.08	**0.40**	0.02
	15-2	**0.73**	0.00	**0.41**	0.00	**0.78**	0.00	0.01	**0.62**	0.13	**0.28**	**0.30**	0.02
	20-2	**0.80**	0.00	**0.53**	0.00	**0.60**	0.00	0.00	**0.67**	0.26	**0.36**	**0.22**	0.01
	Avg	**0.81**	0.00	**0.63**	0.00	**0.78**	0.00	0.12	**0.59**	**0.25**	0.24	**0.31**	0.02
Medium	40-2	**0.98**	0.00	**0.81**	0.00	**0.89**	0.00	0.00	**0.94**	0.06	**0.59**	**0.15**	**0.15**
	40-3	**0.96**	0.00	**0.10**	0.00	**0.91**	0.00	0.00	**0.97**	0.05	**0.65**	**0.12**	0.00
	60-2	**0.57**	0.00	**0.35**	0.02	**0.46**	0.01	0.05	**0.64**	0.20	**0.55**	0.11	**0.26**
	60-3	**0.68**	0.00	**0.63**	0.00	**0.52**	0.00	0.05	**0.81**	0.04	**0.60**	0.20	**0.22**
	80-2	**1.00**	0.00	**0.27**	0.03	**1.00**	0.00	0.00	**0.95**	0.04	**0.71**	**0.23**	0.00
	80-3	**0.99**	0.00	**0.69**	0.00	**0.94**	0.00	0.06	**0.88**	0.23	**0.39**	**0.23**	0.01
	Avg	**0.86**	0.00	**0.48**	0.01	**0.79**	0.00	0.03	**0.87**	0.10	**0.58**	**0.17**	0.11
Large	100-3	**0.95**	0.00	**0.90**	0.00	**0.90**	0.00	0.01	**0.96**	0.11	**0.41**	**0.19**	0.00
	100-5	**0.87**	0.00	**0.84**	0.00	**0.73**	0.00	0.08	**0.82**	0.27	**0.36**	**0.14**	0.10
	150-3	**1.00**	0.00	**0.75**	0.00	**0.99**	0.00	0.00	**0.95**	0.16	**0.54**	**0.15**	0.00
	150-5	**0.92**	0.00	**0.95**	0.00	**0.92**	0.00	0.01	**0.93**	0.21	**0.39**	**0.07**	0.06
	200-3	**1.00**	0.00	**0.40**	0.01	**1.00**	0.00	0.00	**0.92**	0.26	**0.27**	**0.18**	0.00
	200-5	**0.96**	0.00	**0.94**	0.00	**0.96**	0.00	0.00	**0.92**	0.11	**0.42**	**0.12**	0.05
	250-3	**0.98**	0.00	**0.21**	0.01	**1.00**	0.00	0.00	**0.89**	0.24	**0.31**	**0.09**	0.00
	250-5	**0.93**	0.00	**0.84**	0.00	**0.93**	0.00	0.00	**0.87**	0.22	**0.35**	**0.10**	0.04
	Avg	**0.95**	0.00	**0.73**	0.00	**0.93**	0.00	0.01	**0.91**	0.20	**0.38**	**0.13**	0.03

As shown in Table 6, the GD values of the MOABC algorithm outperform those of the NSGA-III, ABC, and PSO algorithms across all 17 instances. This demonstrates the superior convergence and distribution capabilities of the MOABC algorithm. The incorporation of individual hypervolume (HV) values during the onlooker bee phase enables a comprehensive and balanced evaluation of the multiple objectives for each solution, thereby enhancing algorithm's convergence performance.

For the IGD metric, the algorithms are ranked as follows (from best to worst): MOABC, NSGA-III, PSO, and ABC. This ranking is attributed to the advanced neighborhood structure and the six goal-oriented neighborhood search strategies employed by the MOABC algorithm, which facilitate more effective exploration of the search space.

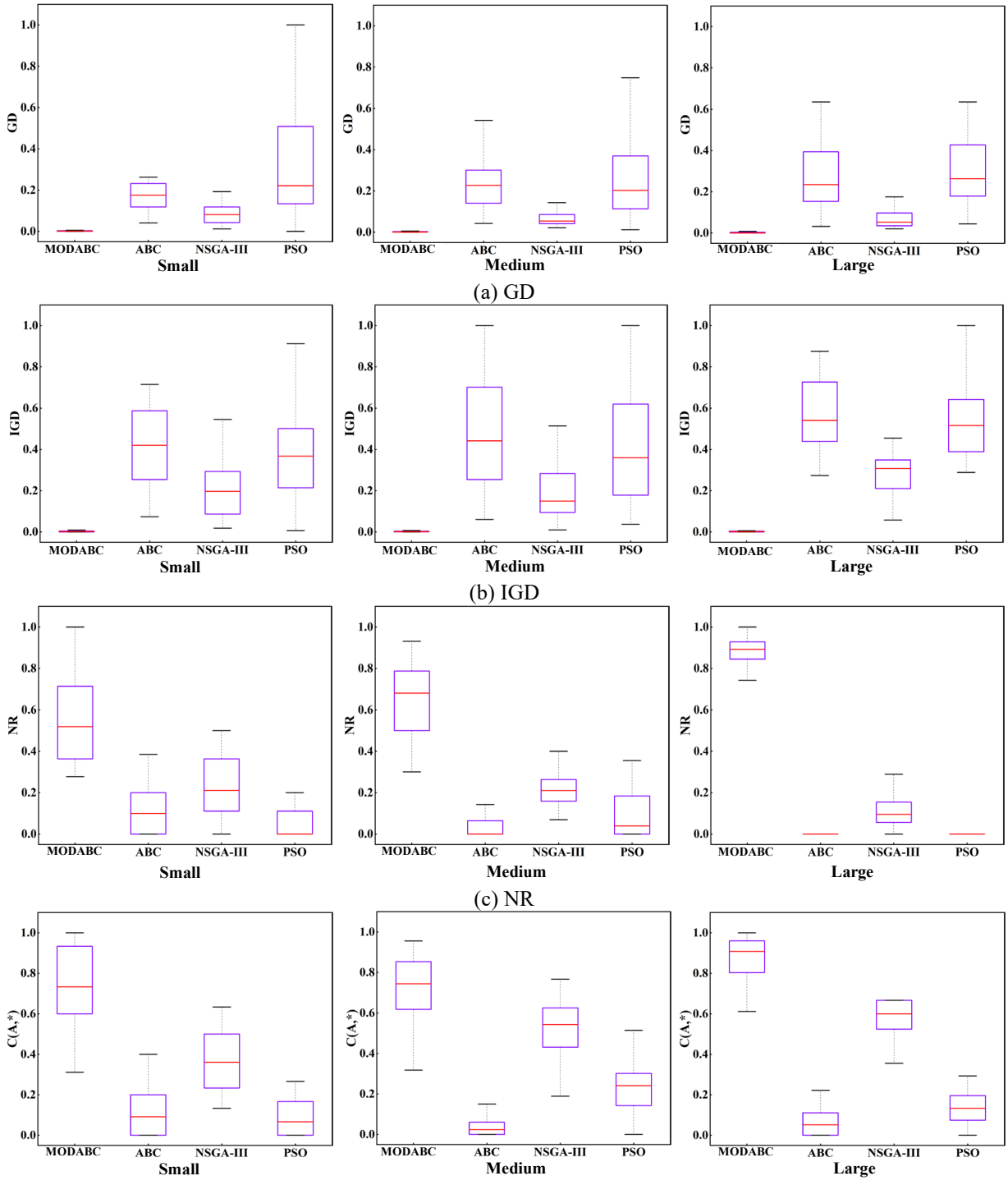
Regarding the NR metric, the MOABC algorithm achieves maximum values of 0.82 and 0.84 for small and medium-sized problems, respectively, and an average value of 0.88 for large-sized problems. This indicates that MOABC algorithm captures over 80% of the potential solution space represented by the NR metric, showcasing its strong performance near the Pareto frontier.

As shown in Table 7, the MOABC algorithm achieves optimal values for the C-matrix metric across all instances. This suggests that the solution set generated by MOABC dominates those of the other algorithms, while rarely being dominated in return. Compared to ABC and PSO, NSGA-III performs better at maintaining population diversity due to its use of reference point rules during the selection stage. However, ABC and PSO, being primarily designed for single-objective optimization, exhibit weaker performance in addressing multi-objective problems.

5.3.2 Robustness comparison

Normalization was applied to address the wide range of values in the GD and IGD metrics. Fig. 8 illustrates box plots of the normalized results for the four algorithms, providing a visual comparison of their performance and highlighting the robustness of their solutions. As shown in Fig. 10(a), the MOABC algorithm achieves the smallest GD values with the highest robustness, maintaining consistency across all three problem sizes. The NSGA-III and ABC algorithms follow in performance, while PSO demonstrates the highest median GD values and the broadest range of variation, indicating lower robustness. In Fig. 10(b), the MOABC also achieves the smallest IGD values with superior robustness across all problem sizes. While NSGA-III performs relatively well, ABC and PSO exhibit larger IGD values and higher variability, reflecting less consistent performance. Fig. 10(c) indicates that the MOABC achieves the highest median NR values, demonstrating superior diversity and convergence compared to the other algorithms. However, for small and medium-sized problems, MOABC also exhibits the

broadest range of variation, resulting in lower robustness compared to NSGA-III. In contrast, the ABC and PSO algorithms have lower NR values but display narrower ranges of variation.



(a) GD

(b) IGD

(c) NR

(d) C-matrix

Fig. 10. Robustness of solutions of different algorithms.

As illustrated in Fig. 10(d), the MOABC algorithm achieves the highest median C-matrix values, indicating better overall performance. However, it also shows the widest range of variation for small and medium-sized problems, leading to the lowest robustness. NSGA-III ranks second in performance, while ABC and PSO demonstrate smaller C-matrix values with narrower ranges of variation. The MOABC algorithm consistently achieves median optimality across all four-evaluation metrics. By leveraging a combination of the PP-FI and FF heuristic, it generates higher-quality initial populations and effectively explores feasible solutions using six neighborhood structures. However, the broader ranges of variation in the NR and C-matrix for small and medium-sized instances suggest that the maximum runtime may not be sufficient for the algorithm to fully converge under these conditions.

5.3.3 Analysis of variance

The Wilcoxon signed rank test, a non-parametric statistical test that does not assume normality or homoscedasticity in the data distribution, was employed to evaluate significant performance differences between the algorithms (Osaba et al., 2021). A p-value of 0.05 or below for any pair of was considered indicative of a statistically significant difference in performance (Ma et al., 2023). As shown in Table 8, the p-values for all experiments involving small, medium, and large-scale problems were significantly below 0.05. This result confirms that the MOABC algorithm demonstrates superior performance in addressing the UPBPM-FPM problem across different problem sizes.

Table 8
The results of the Wilcoxon signed-ranks test

		GD			IGD			NR			C matrix		
	NO.	MOABC ***VS*** ABC	MOABC ***VS*** NSGA-III	MOABC ***VS*** PSO	MOABC ***VS*** ABC	MOABC ***VS*** NSGA-III	MOABC ***VS*** PSO	MOABC ***VS*** ABC	MOABC ***VS*** NSGA-III	MOABC ***VS*** PSO	MOABC ***VS*** ABC	MOABC ***VS*** ABC	MOABC ***VS*** ABC
Small	$R+$	0.0	0.0	1.0	0.0	0.0	0.0	462.0	315.0	406.0	465.0	398.0	465.0
	$R-$	465.0	465.0	465.0	465.0	465.0	465.0	3.00	150.0	59.0	0.0	67.0	0.0
	$p-value$	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	2.31×10^{-6}	3.99×10^{-5}	3.71×10^{-6}	1.73×10^{-6}	8.94×10^{-6}	1.73×10^{-6}
Medium	$R+$	0.0	0.0	0.0	0.0	0.0	0.0	1829.0	1649.0	1711.0	1830.0	1609.5	1826.0
	$R-$	464.0	741.0	741.0	1830.0	1830.0	1830.0	1.0	181.0	119.0	0.0	220.5	4.0
	$p-value$	1.92×10^{-6}	7.74×10^{-8}	7.74×10^{-8}	1.63×10^{-11}	1.63×10^{-11}	1.63×10^{-11}	1.71×10^{-11}	6.36×10^{-11}	3.50×10^{-11}	1.63×10^{-6}	5.29×10^{-9}	1.99×10^{-11}
Large	$R+$	0.0	0.0	0.0	0.0	0.0	0.0	3240.0	3240.0	3240.0	3240.0	2843.0	3240.0
	$R-$	741.0	3240.0	3240.0	3240.0	3240.0	3240.0	0.0	0.0	0.0	0.0	397.0	0.0
	$p-value$	7.74×10^{-8}	7.85×10^{-15}	7.85×10^{-15}	7.85×10^{-15}	7.85×10^{-15}	7.85×10^{-15}	7.83×10^{-15}	7.83×10^{-15}	7.83×10^{-15}	7.84×10^{-15}	6.99×10^{-14}	7.85×10^{-15}

6. Conclusions

This paper addresses the energy-efficient unrelated parallel batch-processing machine scheduling problem with flexible preventive maintenance (UPBPM-FPM), with the goal of minimizing makespan, earliness and tardiness, and total energy consumption. To tackle this complex problem, we propose a novel PP-FI heuristic for generating high-quality initial solutions and develop a MOABC algorithm. The MOABC algorithm integrates the PP-FI and FF heuristics for initial population construction, a hybrid selection strategy that combines the hypervolume index and roulette wheel method to improve diversity and convergence, and a variety of random and goal-oriented neighborhood search methods to refine the Pareto frontier. Experimental results demonstrate the superior performance of the MOABC algorithm over three classical algorithms, NSGA-III, ABC, and PSO, in terms of convergence, diversity, and robustness of the Pareto solutions. The proposed approach successfully balances the trade-offs between multiple objectives, making it highly effective in addressing the challenges posed by flexible preventive maintenance in real-world production environments.

While the study effectively addresses the UPBPM-FPM problem, it does not consider practical constraints such as dynamic job arrivals, sequence-dependent setup times, and predictive maintenance based on machine condition. Future research will focus on extending the MOABC algorithm to incorporate these constraints for more realistic scheduling scenarios. Additionally, exploring the integration of MOABC with dynamic optimization techniques, such as deep reinforcement learning, could further enhance its applicability and performance in complex manufacturing environments.

Acknowledgements

This work has been supported by the National Natural Science Foundation of China, Grant No. 51705370 and Basic Scientific Research Project of Wenzhou City, Grant No. G20240020; G2023036.

Conflict of interest statement

All authors declare that they have no conflict of interest or financial conflicts to disclose.

Data availability statement

The data that support the findings of this study are available from the corresponding author, [M.R.], upon reasonable request.

References

Arroyo, J. E. C., Leung, J. Y. T., & Tavares, R. G. (2019). An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times. *Engineering Applications of Artificial Intelligence*, *77*, 239-254.

Beldar, P., Moghtader, M., Giret, A., & Ansaripoor, A. H. (2022). Non-identical parallel machines batch processing problem with release dates, due dates and variable maintenance activity to minimize total tardiness. *Computers & Industrial Engineering*, *168*, 108135.

Fowler, J. W., & Mönch, L. (2022). A survey of scheduling with parallel batch (p-batch) processing. *European Journal of Operational Research*, *298*(1), 1-24.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Huang, J., Wang, L., & Jiang, Z. (2020). A method combining rules with genetic algorithm for minimizing makespan on a batch processing machine with preventive maintenance. *International Journal of Production Research*, *58*(13), 4086-4102.

Jang, J. W., Kim, Y. J., & Kim, B. S. (2022). A Three-Stage ACO-Based Algorithm for Parallel Batch Loading and Scheduling Problem with Batch Deterioration and Rate-Modifying Activities. *Mathematics*, *10*(4), 657.

Jia, Z.-h., Huo, S.-y., Li, K., & Chen, H.-p. (2020). Integrated scheduling on parallel batch processing machines with non-identical capacities. *Engineering Optimization*, *52*(4), 715-730. https://doi.org/10.1080/0305215X.2019.1613388

Jia, Z.-h., Li, K., & Leung, J. Y. T. (2015). Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities. *International Journal of Production Economics*, *169*, 1-10.

Jia, Z.-h., Zhang, Y.-l., Leung, J. Y. T., & Li, K. (2017). Bi-criteria ant colony optimization algorithm for minimizing makespan and energy consumption on parallel batch machines. *Applied Soft Computing*, *55*, 226-237.

Li, K., Zhang, H., Chu, C., Jia, Z.-h., & Chen, J. (2022). A bi-objective evolutionary algorithm scheduled on uniform parallel batch processing machines. *Expert Systems with Applications*, *204*, 117487.

Li, M., & Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys (CSUR)*, *52*(2), 1-38.

Ma, Z., Wu, G., Suganthan, P. N., Song, A., & Luo, Q. (2023). Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms. *Swarm and Evolutionary Computation*, *77*, 101248.

Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., Suganthan, P. N., Coello, C. A. C., & Herrera, F. (2021). A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm and Evolutionary Computation*, *64*, 100888.

Rauf, M., Guan, Z., Yue, L., Guo, Z., Mumtaz, J., & Ullah, S. (2020). Integrated planning and scheduling of multiple manufacturing projects under resource constraints using raccoon family optimization algorithm. *IEEE Access*, *8*, 151279-151295.

Suhaimi, N., Nguyen, C., & Damodaran, P. (2016). Lagrangian approach to minimize makespan of non-identical parallel batch processing machines. *Computers & Industrial Engineering*, *101*, 295-302.

Wang, Y., Jia, Z.-h., & Li, K. (2021). A multi-objective co-evolutionary algorithm of scheduling on parallel non-identical batch machines. *Expert Systems with Applications*, *167*, 114145.

Xiao, X., Ji, B., Yu, S. S., & Wu, G. (2024). A tabu-based adaptive large neighborhood search for scheduling unrelated parallel batch processing machines with non-identical job sizes and dynamic job arrivals. *Flexible Services and Manufacturing Journal*, *36*(2), 409-452.

Yuan, Y., Xu, H., & Wang, B. (2014). An improved NSGA-III procedure for evolutionary many-objective optimization.

Yue, L., Guan, Z., Zhang, L., Ullah, S., & Cui, Y. (2019). Multi objective lotsizing and scheduling with material constraints in flexible parallel lines using a Pareto based guided artificial bee colony algorithm. *Computers & Industrial Engineering*, *128*, 659-680.

Zeng, C., & Liu, J. (2024). A metaheuristic algorithmic framework for solving the hybrid flow shop scheduling problem with unrelated parallel machines. *Engineering Optimization*, 1-24. https://doi.org/10.1080/0305215X.2024.2372634

Zhang, C.-L., Fan, G.-Q., & Wang, J.-Q. Fair scheduling on parallel batch machines with non-identical capacities. *Engineering Optimization*, 1-15. https://doi.org/10.1080/0305215X.2024.2351188

Zhang, H., Wu, F., & Yang, Z. (2021). Hybrid approach for a single-batch-processing machine scheduling problem with a just-in-time objective and consideration of non-identical due dates of jobs. *Computers & Operations Research*, *128*, 105194.

Zhang, H., Yang, Y., & Wu, F. (2022). Just-in-time single-batch-processing machine scheduling. *Computers & Operations Research*, *140*, 105675.

Zhang, J., Yao, X., & Li, Y. (2020). Improved evolutionary algorithm for parallel batch processing machine scheduling in additive manufacturing. *International Journal of Production Research*, *58*(8), 2263-2282.